DYNAMIC RESOURCE LOCATION IN PEER-TO-PEER NETWORKS

A Thesis

by

RIPAL BABUBHAI NATHUJI

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

May 2003

Major Subject: Computer Engineering

DYNAMIC RESOURCE LOCATION IN PEER-TO-PEER NETWORKS

A Thesis

by

RIPAL BABUBHAI NATHUJI

Submitted to Texas A&M University
in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

Approved as to style and content by:

Riccardo Bettati (Chair of Committee)	Wei Zhao (Member)
Narasimha Reddy (Member)	Valerie Taylor (Head of Department)

May 2003

Major Subject: Computer Engineering

ABSTRACT

Dynamic Resource Location in Peer-to-Peer Networks. (May 2003)
Ripal Babhubhai Nathuji, B.S., Massachusetts Institute of Technology
Chair of Advisory Committee: Dr. Riccardo Bettati

Resource location is a necessary operation for computer applications. In large scale peer-to-peer systems, random search is a scalable approach for locating dynamic resources. Current peer-to-peer systems can be partitioned into those which rely upon the Internet for message routing and those which utilize an overlay network. These two approaches result in different connectivity topologies. This thesis analyzes the effect of topological differences on the effectiveness of random search. After demonstrating the benefits of an overlay network, we propose a hybrid approach for resource location. Our proposed protocol provides deterministic searching capabilities which can help prevent request failures for sensitive applications.

TABLE OF CONTENTS

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

FIGURE Page

CHAPTER I

INTRODUCTION

The Internet is a powerful infrastructure for computer applications. By establishing a large scale network of heterogeneous nodes, it provides ample resources for computing systems. Distributed programs and agile objects are just a few examples of the types of applications that take advantage of such an environment.

Locating resources becomes problematic in large networks. Centralized schemes that are effective for small networks scale poorly. When resources are dynamic in nature, a centralized approach becomes even less plausible. Thus as systems increase in size, it becomes clear that a peer-to-peer approach is needed in place of a centralized architecture.

A possible alternative to a centralized architecture is the use of a random algorithm. Randomized load balancing schemes are analyzed in [1]. The author presents a model termed the *supermarket model* in which jobs are assigned randomly between n servers. In particular, jobs pick d processors with a uniform probability, and are queued to the processor which is least loaded.

This cheap approach for load balancing works well because jobs choose randomly among all processors. Unfortunately, in peer-to-peer systems, all peers are possible servers for a request. Thus a direct mapping of the algorithm would require nodes to maintain a global membership list. Handling this type of state obviously does not scale in peer-to-peer systems. It is therefore important to determine how well simple random algorithms can actually perform when peers only have partial membership information. The peer-to-peer systems we consider only require partial membership

---

The journal model is *IEEE Transactions on Automatic Control.*

state at nodes.

In a peer-to-peer architecture, random search can be performed by forwarding requests to neighbors randomly until either the resources are found or the time-to-live (TTL) for the request expires. By making forwarding decisions based on local neighbor information, the algorithm can be completely distributed and scalable.

Current peer-to-peer systems subscribe to one of two mechanisms for message routing. Some architectures use the underlying Internet topology for routing. These systems forward messages to peers by using normal IP routing. Peer-to-peer systems that use this approach are characterized by power-law distributions in their node connectivity.

The remaining peer-to-peer designs use overlay networks to provide theoretical bounds for message routing overhead. In these systems messages are routed to the destination by other peers. These overlay networks can be manipulated to provide a fully connected topology graph for random search algorithms. This topological difference between peer-to-peer systems that utilize overlay networks and those that do not raises the question of the relationship between the underlying topology of networks and the effectiveness of random search.

Though random search has been found to be effective in terms of overhead and load balancing, it is an unattractive alternative for some applications. Since random algorithms employ a TTL on requests to limit overhead, requests may fail even when resources are available in the system. It would be beneficial to provide an alternative means of resource location for applications which cannot tolerate such failures. In particular, a deterministic extension to random search would help address this problem.

The remainder of this document describes research which relates to the problem of dynamic resource location in peer-to-peer networks. Chapter II presents previous

work and background information. It is followed by the thesis work of the author. The thesis can be divided into a two part research effort. The first segment of work is presented in Chapters III and IV. This segment concentrates on analyzing the topological effects of candidate peer-to-peer architectures on a random search algorithm. The analysis is conducted using analytical models which are validated with simulation results. The analysis will show that the use of an overlay network to obtain a fully connected topology is beneficial for random search.

The second segment of research is presented in Chapter V. It proposes a modified protocol that supports deterministic search capabilities for critical applications. In particular, a hybrid overlay network architecture is presented. This architecture will allow for both random and deterministic searching capabilities. A summary and final conclusions of the work are presented in Chapter VI.

CHAPTER II

RELATED WORK

Peer-to-peer networks are a growing area of research in computer science. A variety of architectures have been designed for these networks. In this chapter, current peer-to-peer systems are introduced. Also, existing research in peer-to-peer resource location and power-law network topologies is presented.

As previously mentioned, peer-to-peer systems can be divided into two subgroups. Some peer-to-peer systems rely on the Internet for message routing, and have a power-law distribution in their node connectivity. Examples of such systems are Freenet [2] and Gnutella [3, 4]. These architectures are presented in section A. The remaining peer-to-peer systems employ overlay networks for message routing. Chord [5], Pastry [6], and CAN [7] fall into this category. In particular, these systems use a hash based mechanism to establish and route in the overlay network. These architectures are presented in section B.

The problem of resource discovery in peer-to-peer networks has been handled in previous research. INS/Twine [8],[9] is a scalable architecture for resource discovery. We present this system in section C and explain why it cannot be used for dynamic resources. Finally, the chapter concludes by explaining the various alternatives for random resource discovery in peer-to-peer networks.

A. Peer-to-Peer Power Law Systems

1. Gnutella

Gnutella, as with all of the peer-to-peer systems that are presented, is a protocol that is used extensively for file sharing. Its messaging protocol supports the dynamic

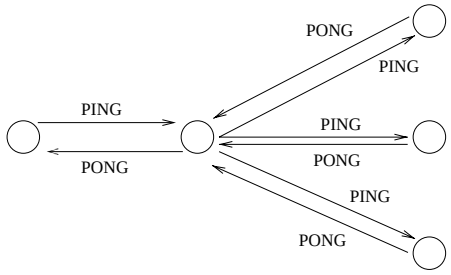

Fig. 1. Gnutella Membership Join Protocol

environment that results in peer-to-peer networks.

Nodes join a Gnutella network using special group membership messages. New members first contact some well known node which has already obtained membership. This node forwards a PING message on behalf of the initiator to all of its neighbors, while returning a PONG message with its IP address and other relevant information. Subsequent nodes which receive the PING message also forward the request and back-propagate their own PONG message. This process continues until all PING messages have exceeded some TTL. The joining node then creates TCP connections with as many neighbors as it desires. This process is displayed in Figure 1.

Searches in the Gnutella network behave similar to membership initiation. A query initiator forwards requests to all of its known neighbors. The query is continually forwarded until some TTL is exceeded. All nodes which can respond to the query reply back to the initiator. Thus Gnutella uses a request flooding technique to handle queries.

2. Freenet

Freenet is another peer-to-peer system that is used for information storage and retrieval. Freenet uses keys to identify nodes in the system. Nodes are assigned their

respective keys during the joining process.

As with Gnutella, nodes join the Freenet network by cooperating with a well known node that is already a member. In order to provide security, nodes cannot determine their own keys. Instead, when joining the network a node includes the hash of a random seed with an announcement message. The same hash function is used by all nodes in the system.

When a node receives an announcement message, it creates its own random seed and XOR's it with the received hash value. The hash of this XOR is the commitment value for the node. The new hash is then forwarded to some random neighbor, and the process continues until the TTL of the request is exceeded. The last node to receive the request only generates a random seed. Finally all nodes that participated in the chain reveal their seeds. The XOR of the seeds is used as the key value for the new node, and the commitment values are used to confirm that all nodes reveal their seeds truthfully.

Information is retrieved from the Freenet system using binary file keys that are associated with the target objects. These keys are generated using descriptive strings for the object that is being searched for. Nodes forward the request by searching their routing table for the neighbor which has a key value closest to that of the request. A second-nearest, or third-nearest, and so on may be used to avoid creating forwarding loops. The request ends when a node with the object is reached, or the TTL of the request is exceeded. Thus Freenet employs a random-walk approach for data retrieval.

3. Power-Law Distributed Topologies

The Internet is a network which was widely believed to have random characteristics in its topology. It has recently been shown that the topology of the Internet actually follows a power-law distribution [10]. This distribution can be described by the gen-

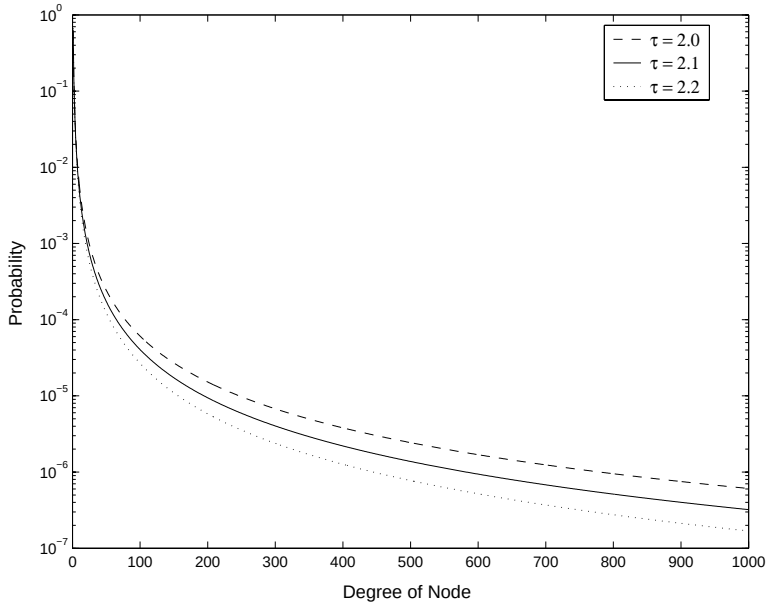

Fig. 2. Power-Law Node Connectivity Distributions

erating function in equation (2.1), [11]. In this equation k is the degree of a node, m is the maximum degree of any node in the network, and τ is a parameter that characterizes the power-law distribution.

$$G_0(x) = c\sum_{k=1}^{m} k^{-\tau}x^k \tag{2.1}$$

Figure 2 illustrates the probability distribution for various values of τ. Large networks usually have ($2 \leq \tau \leq 2.3$) [11]. This power-law distribution in node connectivity has a simple interpretation. Most nodes in the network have low connectivity, and a small set of "well known nodes" have high connectivity. Measurements of Freenet, Gnutella, and other peer-to-peer topologies have revealed that node connectivity in these networks also follow power-law distributions [2], [4],[11], [12].

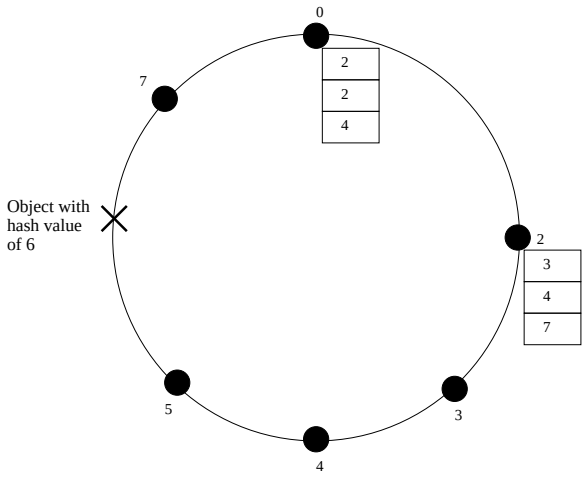

Fig. 3. Chord Architecture

B. Peer-to-Peer Overlay Networks

1. Chord and Pastry

Chord and Pastry are hash based peer-to-peer systems which use a ring based structure as an overlay network. The two architectures differ in the precise routing algorithms and routing tables used. Chord requires that nodes have a routing table of size $O(log_2 N)$ in an N-node system, and routes messages using $O(log_2 N)$ nodes. Pastry uses a routing table of size $O(log_{2^b} N)(2^b - 1)$ where b is a configured parameter. Messages can be routed in Pastry using $O(log_{2^b} N)$ nodes.

Since Chord and Pastry share similar architectural aspects and benefits, only the details of the Chord system are described here. Chord uses a key system to identify nodes and query objects. Nodes in the system generate their node IDs by hashing a unique attribute such as their IP address. Nodes are then organized in the overlay network using a ring structure where nodes are linked with increasing node ID values. Every node has a pointer to its successor in the ring, as well as its predecessor. The

successor of the node with the largest ID is the node with the smallest ID, thus forming the ring.

Since a hash function is used to place nodes in the Chord ring, probabilistically, the key space is divided into equal partitions. Objects are assigned to nodes in a simple fashion. The descriptor for the object is hashed giving a value in the ring key space. The object is then assigned to the successor of the point in the key space that is hashed to. For example, the object which hashes to an ID value of 6 in Figure 3 would be assigned to the node with ID 7.

Using only the predecessor and successor pointers, the Chord ring can only perform linear search. In order to obtain the $O(log_2 N)$ performance, nodes must use a specifically structured routing table. The routing table consists of m entries, where the i^{th} entry in the table at node n is the first node which succeeds the value $(n+2^{i-1})$. The value of m is the number of bits in the node identifiers. In our example of Figure 3, $m = 3$. The figure displays the routing tables for nodes 0 and 2.

Using these routing tables, nodes solve queries by forwarding requests to the node with the highest ID which is less than the key of the request. This continues until a node n receives the request such that $n < key \leq n.successor$. At this point the request is finally routed to $n.successor$.

2. CAN

CAN, or Content-Addressable Network, is another hash based peer-to-peer system. As opposed to a ring based structure, CAN uses a geometric approach to organizing nodes. Nodes are assigned positions in a d-dimensional space. This allows CAN to conduct message routing using $O(N^{\frac{1}{d}})$ nodes.

In order to organize nodes and assign objects for data sharing, CAN uses d hash functions. A node uses these hash functions to obtain d key space values for itself.

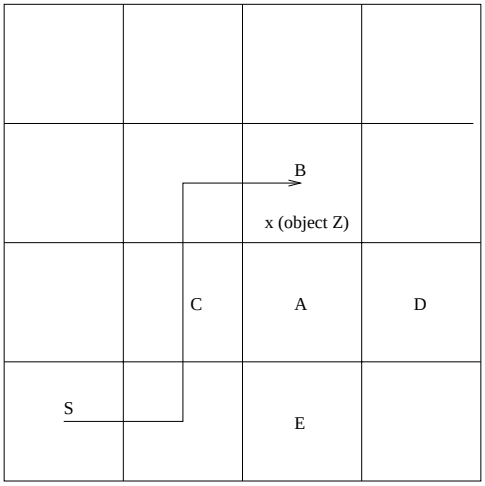

Fig. 4. CAN Routing

These values correspond to a point in the d-dimensional space. Nodes cooperate with each other to divide up the d-dimensional space given their own positions in the space. Figure 4 illustrates an example in two-dimensional space. Again, probabilistically, this should result in an equally partitioned space. Every node keeps $2d$ pointers such that they have a successor and predecessor in every dimensions. These successors and predecessors are defined according to the spatial partitioning. So, for example, in Figure 4 node A has neighbors C and D in one dimension, and B and E in the other.

In order to locate objects in the system, the descriptor for the object is hashed with the d hash functions. Nodes then forward the request to the neighbor which is geometrically closest to the point in space described by the resulting key values. This continues until the request is received by the node which controls the space that the object keys correspond to. Figure 4 illustrates this process. The request initiates as a request for object Z at node S. The request is forwarded until it is received by node B which has the object.

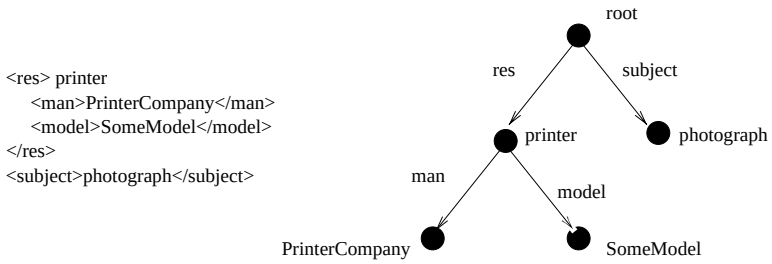

Fig. 5. INS Resource Descriptor and AVTree

3. Fully Connected Topologies

Peer-to-peer systems which use a hash based approach can be manipulated to provide a fully connected topology for random search algorithms. In the Chord and Pastry systems, hash functions distribute nodes equally across the ring overlay structure. Similarly, the hash based approach in CAN divides the d-dimensional space into partitions with equal volume.

By generating a random key in the ring, or a random point in the CAN key space, a node can pick a globally random node to forward requests to. For Chord and Pastry, this can be done by hashing a random number generated at the node. Similarly, CAN nodes can pick a random node by hashing a random number with the d hash functions used in the system. Thus, using these peer-to-peer overlay networks, a random algorithm can utilize a fully connected topology without the cost of complete topological knowledge at every node.

C. Peer-to-Peer Resource Discovery

INS/Twine is a peer-to-peer resource discovery protocol. It relies upon attribute descriptive strings for locating resources. Thus it is capable of handling a large set of heterogeneous resources in a scalable fashion.

In order to provide intentional resource discovery, INS/Twine converts resource descriptions into attribute-value pairs (AVTree). Figure 5 displays an example of a descriptor to AVTree conversion for a photo quality printer. In INS/Twine, a query matches a resource if the AVTree of the request matches some sub tree of the resource AVTree.

In order to distribute resource information, INS/Twine designates some nodes as special resolver nodes. These nodes use a Chord ring (or some other hash based peer-to-peer infrastructure) to coordinate. In order to advertise a resource, the AVTree describing the resource is divided into strands. Each strand is hashed and assigned to a resolver in the underlying Chord ring. The resolver keeps an association between the strand and the location of the resource.

Resource location is handled in a similar fashion. The AVTree of the request is parsed into strands, hashed, and sent to the appropriate resolvers. The results of any matching strands are returned to the requestor. Thus many possible resource sources may be returned from a single request.

Though INS/Twine provides a scalable peer-to-peer infrastructure for locating resources, it is not a valid solution for dynamic resources. This is due to the fact that resolvers in the Twine architecture contain state regarding resources. Though this state is soft, and can handle infrequent changes to resource descriptors, it is not designed to handle the frequent updates which result with dynamic resources.

D. Random Resource Discovery Algorithms

It is been suggested in [13] that random walks are a more efficient approach for searching in peer-to-peer networks than request flooding. This is due to the large amounts of traffic that can be generated by flooding techniques. Therefore we will

assume a random walk approach for our resource location algorithms.

In order to simplify analysis, we will assume that the resource location protocol will make use of a single random walk. This random walk is conducted by random forwarding at every node. The only constraint for forwarding is that the request is forwarded to a node that is not the current node.

Given the peer-to-peer architectures that we've discusses, there are two types of forwarding that we need to consider. The first is based on peer-to-peer networks which have a power-law distributed topology. For these systems nodes forward requests to a neighbor in their local routing table. The second approach is for systems which use a hash based overlay architecture. In these systems, nodes forward requests to a globally random node by hashing a random number. Thus, given a particular topology, we have an appropriate random location algorithm.

CHAPTER III

ANALYTICAL MODELS

In order to analyze the effect of topology on random resource location, we develop analytical models. This chapter describes the analytical model for the two types of topologies that we consider. Section A describes the analytical node models that are used. Section B presents the model and solution for random resource location in fully connected networks. The chapter concludes with the model for power-law distributed networks in section C.

A. Node Model

In order to develop our analytical models, we must first identify the parameters of the problem at hand. As discussed at the end of Chapter II, we assume a random walk approach for random resource location. For the random walk, we assume that a request can be forwarded a maximum of K times (TTL=K).

We are concerned with the effect of topology on random resource location. In order to concentrate on this aspect of the problem, we do not want to have to consider the particular distribution of available resources in the system, or the distribution of the types of resource requests that are generated. Therefore, we make some simplifying assumptions. In particular, we assume that a node can only serve a single request at a time. Subsequent requests are queued up at the node. In our model, nodes have a queue of length M where the request at the front of the queue is the current active request at the node.

Requests have service times which are modeled as exponential distributions with mean $\frac{1}{\mu}$. Arriving requests at nodes are modeled as Poisson processes with rates λ_0. We will show that with our model forwarded requests that are received at a node

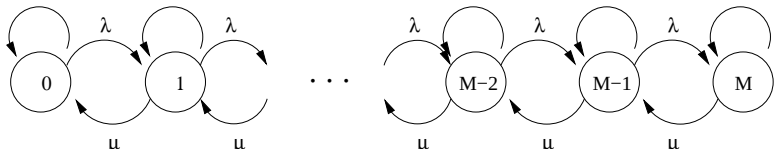

Fig. 6. Node Markov Model

are also Poisson processes. Assigning rates $\lambda_1, \lambda_2, \ldots, \lambda_K$ for forwarded requests, the total rate λ for incoming requests is given by equation (3.1). The resulting node Markov chain used for our model is the M/M/1 queue shown in Figure 6.

$$\lambda = \sum_{i=0}^{K} \lambda_i \tag{3.1}$$

Let p_j be the probability that a request arrives when the node is in state j. We are particularly interested in p_M, since requests that arrive from the Poisson streams $\lambda_0, \lambda_1, \ldots \lambda_{K-1}$ are forwarded with this probability. Moreover, requests that arrive from the Poisson stream λ_K are dropped with probability p_M. These dropped requests are considered failures for our analysis. Asymptotically, $p_j = (\frac{\lambda}{\mu})p_{j-1}$. Therefore $p_M = (\frac{\lambda}{\mu})^M p_0$. The closed form for p_M as a function of λ is given by equation (3.2).

$$p_M = F(\lambda) = \frac{(\frac{\lambda}{\mu})^M}{\sum_{k=0}^{M}(\frac{\lambda}{\mu})^k} \tag{3.2}$$

Poisson processes can be decomposed into individual Poisson streams. Given the probability p_M, the stream of new requests λ_0 can be decomposed into two streams of rate $\lambda_0(1 - p_M)$ and $\lambda_0 p_M$. The first of the decomposed streams is a Poisson process of requests that are served or queued at the node, and the latter stream is a Poisson process of requests that are forwarded for the first time. This analysis can be repeated for the streams $\lambda_1, \lambda_2, \ldots, \lambda_{K-1}$ to show that forwarded requests can be modeled as Poisson processes. Figure 7 illustrates the incoming and forwarded Poisson streams

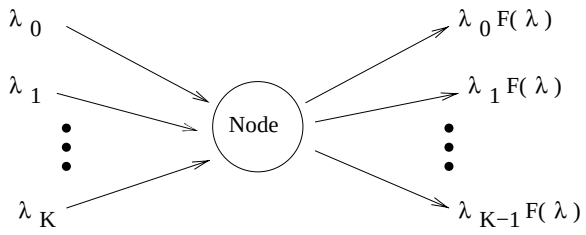

Fig. 7. Request Forwarding Streams

at a node.

B. Fully Connected Networks

In order to analyze random resource location in fully connected networks, we analyze a more general model. In particular, we begin by assuming that all nodes have the same degree. Fully connected networks are a special case of this model where all nodes have degree $(N-1)$.

When all nodes have the same degree, they have the same rate of incoming requests $\lambda_0, \lambda_1, \ldots, \lambda_K$. Figure 7 shows that every node will have request forwarding streams $\lambda_0 F(\lambda), \lambda_1 F(\lambda), \ldots, \lambda_{K-1} F(\lambda)$. Since all nodes have the same degree D, with random forwarding the stream $\frac{\lambda_i F(\lambda)}{D}$, $1 \leq i \leq (K-1)$, is received from each neighbor of a node. Subsequently, each node has a total incoming forwarding stream of $D(\frac{\lambda_i F(\lambda)}{D}) = \lambda_i F(\lambda)$. This analysis gives us the system of nonlinear equations given in equation (3.3).

$$F(\lambda)\lambda_{j-1} = \lambda_j, 1 \leq j \leq K \tag{3.3}$$

The system of equations which describe the model can be solved using a recursive method. Starting with λ_0 and $\lambda_1 = \lambda_2 = \ldots = \lambda_K = 0$, $F(\lambda)$ is calculated. The values for $\lambda_1, \lambda_2, \ldots, \lambda_K$ are then recalculated using equation (3.3). This process is

repeated until equation (3.4) is minimized beyond some threshold. The values of $\lambda_1, \lambda_2, \ldots, \lambda_K$ thereby completely characterize the fully connected network model.

$$F_{FullMin}(\lambda_0, \lambda_1, \ldots, \lambda_K) = \sum_{j=1}^{K} (F(\lambda)\lambda_{j-1} - \lambda_j)^2 \tag{3.4}$$

The performance criteria that will be used for our models are the probability of request failure, and the rate of request failures. The probability of failure and the failure rate for fully connected networks is given by equation (3.5).

$$P(FullRequestFailure) = \frac{\lambda_K F(\lambda)}{\lambda_0} = \frac{\lambda_{FullRequestFailureRate}}{\lambda_0} \tag{3.5}$$

C. Power-Law Distributed Networks

In power-law distributed networks, a node has degree D such that $degree_{min} \leq D \leq degree_{max}$. For our analysis, we assume $degree_{min} = 1$ and $degree_{max} = m$. The generating function for the connectivity distribution is therefore given by equation (2.1).

Given a random link in the power-law distributed network, we can obtain the probability distribution for the degree of the node on the other side of the link. This distribution is described by the generating function in equation (3.6) obtained from [11].

$$G_1(x) = \frac{\sum_{k=1}^{m} ck^{-\tau+1}x^k}{\sum_{k=1}^{m} ck^{-\tau+1}} \tag{3.6}$$

Let A be the event that node N_1 has degree K_1. Given this information, we would like to know the probability distribution for degree D of N_1's neighbors. Equation (3.7) can be obtained using conditional probability. Without assuming some structure for the conditional probabilities $P(D|A)$,$P(A|D)$, we must assume that the events A and D are independent giving the result in equation (3.8). Therefore, the probability

distribution for the degree of N_1's neighbors at each outgoing link is given by (3.6).

$$P(D|A) = \frac{P(D \cap A)}{P(A)} = \frac{P(A|D)P(D)}{P(A)} \tag{3.7}$$

$$P(D|A) = \frac{P(D \cap A)}{P(A)} = \frac{P(D)P(A)}{P(A)} = P(D) \tag{3.8}$$

Since nodes may have different degrees in power-law distributed networks, they may have different rates of incoming requests. For fully connected networks λ_j denoted the incoming rate of requests which have been forwarded j times. In our power-law model, we have $\lambda_{j,k}$ as the rate of requests which have been forwarded j times and are being forwarded by a node with degree k. This leads us to a per link distribution for the rate of incoming requests which have been forwarded j times on each link. The generating function for this distribution is given by equation (3.9). Note that the rate $\lambda_{j,k}$ is divided by k since the forwarded requests are distributed randomly among k links.

$$G_2(x) = \frac{\sum_{k=1}^{m} ck^{-\tau+1} x^{\frac{\lambda_{j,k}}{k}}}{\sum_{k=1}^{m} ck^{-\tau+1}} \tag{3.9}$$

We can now derive the distribution for the total rate of incoming requests at a node of degree d. The generating function for the arrival of requests which have been forwarded j times is given by equation (3.10).

$$G_3(x) = \left(\frac{\sum_{k=1}^{m} ck^{-\tau+1} x^{\frac{\lambda_{j,k}}{k}}}{\sum_{k=1}^{m} ck^{-\tau+1}} \right)^d \tag{3.10}$$

To calculate values for our performance criteria, we need to use the expected value of incoming requests for nodes. Let $f(j,d)$ denote the expected value for the rate of requests which have been forwarded j times that arrive at a node of degree d. This expected value is simply $G_3'(1)$ for the appropriate value of j and d. Equation

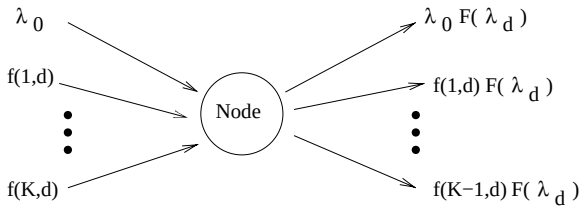

Fig. 8. Forwarding Streams for Power-Law Model

(3.11) defines $f(j,d)$.

$$f(j,d) = \frac{(d)\sum_{k=1}^{m} k^{-\tau}\lambda_{j,k}}{\sum_{k=1}^{m} k^{-\tau+1}} \tag{3.11}$$

Using the definition of $f(j,d)$, we can redefine the total incoming request rate λ for our power-law distributed model. In this model, λ depends on the degree of the node. Therefore, we denote λ_d as the total rate of incoming requests at a node of degree d. This rate is defined in equation (3.12).

$$\lambda_d = \lambda_0 + \sum_{j=1}^{K} f(j,d) \tag{3.12}$$

Similar to the model for fully connected topologies, we define $F(\lambda_d)$ as the probability that a request arrives at a node whose queue is full. Figure 8 displays the altered forwarding request streams for the power-law topology model. This result gives us the set of nonlinear equations to model the system. The system of equations is given in equations (3.13a)-(3.13b).

$$F(\lambda_d)\lambda_0 = \lambda_{1,d}, 1 \leq d \leq m \tag{3.13a}$$

$$F(\lambda_d)f(j-1,d) = \lambda_{j,d}, 2 \leq j \leq K, 1 \leq d \leq m \tag{3.13b}$$

This system of equations can be solved by mapping it to a minimization problem as was done for the fully connected topology model. The minimization function is

given by equations (3.14a)-(3.14c). Using recursion, the values $\lambda_{j,d}, (1 \leq j \leq K), (1 \leq d \leq m)$ can be obtained from the minimization function.

$$\sum_{d=1}^{m} \Big(F(\lambda_d)\lambda_0 - \lambda_{1,d} \Big)^2 \tag{3.14a}$$

$$\sum_{d=1}^{m} \sum_{j=2}^{K} \Big(F(\lambda_d) f(j-1,d) - \lambda_{j,d} \Big)^2 \tag{3.14b}$$

$$F_{PowerMin} = (3.14a) + (3.14b) \tag{3.14c}$$

In order to complete our model for random resource location in power-law distributed networks, we must define the formulas for our performance criteria. The formula for overall request failure probability is given by equation (3.15), where c is the normalization constant for equation (2.1). Request failure rates are analyzed per each class of node degree. Equation (3.16) defines the failure rate as a function of node degree.

$$P(PowerRequestFailure) = \frac{\sum_{d=1}^{m} c d^{-\tau} f(K,d) F(\lambda_d)}{\lambda_0} \tag{3.15}$$

$$\lambda_{PowerRequestFailureRate}(d) = f(K,d) F(\lambda_d) \tag{3.16}$$

CHAPTER IV

RESULTS

This chapter presents the results of our analytical models. Simulation data is also presented in order to validate our theoretical results. Section A describes the simulation environment and parameters used. Section B provides results and explanations of our findings. Conclusions regarding the results are discussed in section C.

A. Simulation Environment

In order to simulate resource location in peer-to-peer networks, the simulation package JavaSim was used [14]. For the power-law distributed peer-to-peer network simulations, the Inet topology generator was used [15]. Simulations used 5000 nodes, a queue size of 10, a mean service time of 1, and a maximum degree of 1025. The maximum degree for our nodes was constrained by the topology generator.

Simulations for the fully connected network allowed nodes to forward requests randomly to any node except for themselves. In the power-law topology network requests were forwarded randomly to some neighbor. When generating the topologies, a value of 2.1 was used for τ. Though this value of τ closely models the topology of the Internet, it may not model all peer-to-peer networks as accurately. Our results will show that our conclusions are independent of the exact value of τ which models a given peer-to-peer network. Figure 9 compares the power-law connectivity characteristics of a simulation topology to the theoretical distribution where $\tau = 2.1$ and the network has 5000 nodes.

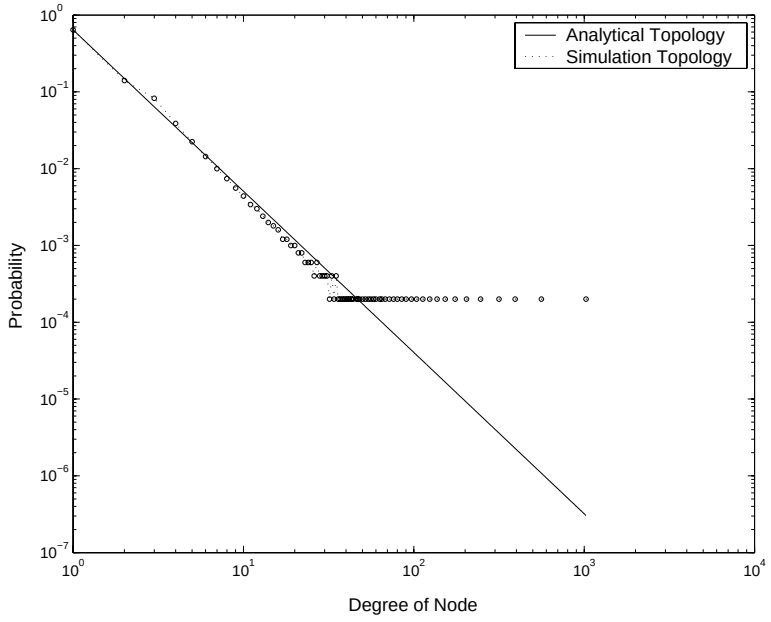

Fig. 9. Simulation vs. Theoretical Topology Distribution

B. Analytical and Simulation Results

1. Analytical Results

Our analytical models were solved for various values of $\frac{\lambda_0}{\mu}$. In order to verify that the model expressed expected trends, we obtained results for networks with TTL set to 10, 5, and 1. The remaining model parameters were configured as discussed in section A.

Figure 10 displays results for nodes with a request TTL of 10. In the fully connected network, there are no failures until the network reaches near one hundred percent utilization. The power-law network, on the other hand, exhibits significant request failures compared to the fully connected network scenario.

Figures 11 and 12 give the analytical results when requests have time-to-live values of 5 and one respectively. They support the same relationship for request

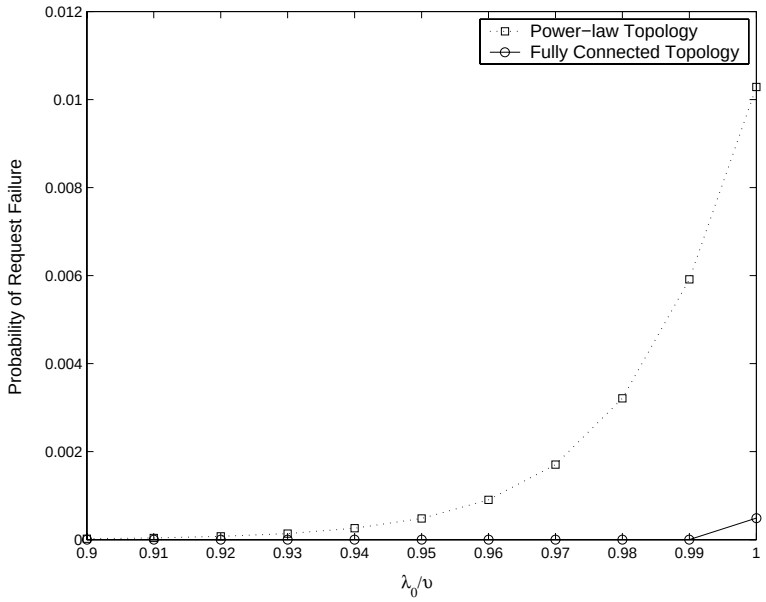

Fig. 10. Analytical Failure Probability with TTL=10

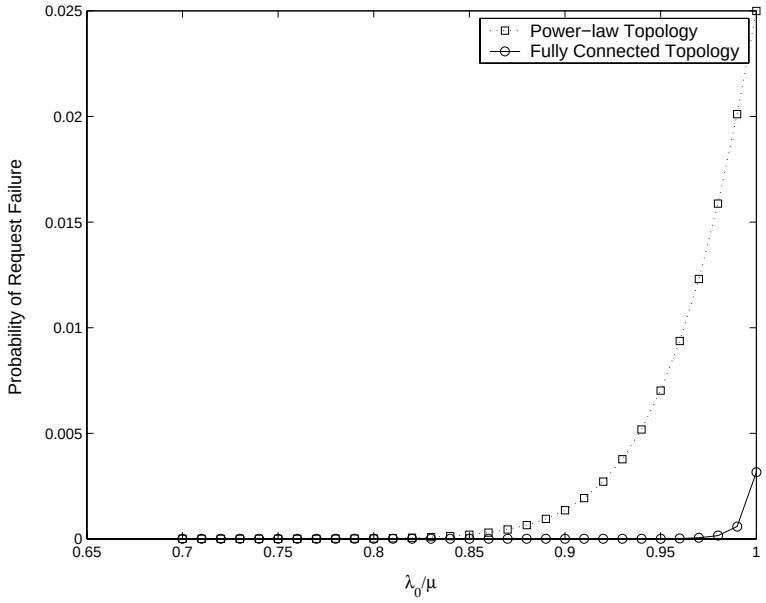

Fig. 11. Analytical Failure Probability with TTL=5

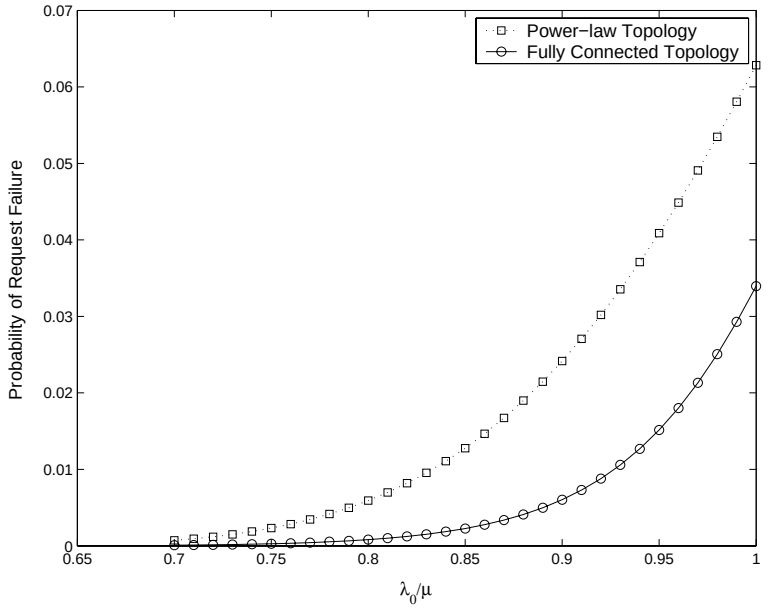

Fig. 12. Analytical Failure Probability with TTL=1

failure probabilities between power-law networks and fully connected networks found in the case of request TTL set to 10. Moreover, as expected, the overall failure probabilities increase in both networks as the request TTL is reduced.

Figure 13 displays the request failure rates that occur in the power-law analytical model with $\lambda_0 = 0.95$. The results show that failure rates increase with increasing node degree. Table I compares the failure rates of nodes in the fully connected network model to single degree nodes in the power-law model. The table shows that nodes in the fully connected network experience lower failure rates than single degree nodes in the power-law network with TTLs of 5 or 10, but a higher rate with a TTL of 1.

Table II provides the ratio of failure rates given in Table I. It is apparent that the ratio of failure rates grows quickly with increasing TTL. In a power-law network with $\tau = 2.1$, 64% of nodes have a degree of one. In spite of this, when the request TTL is one, poorly connected nodes have a smaller failure rate than nodes in the

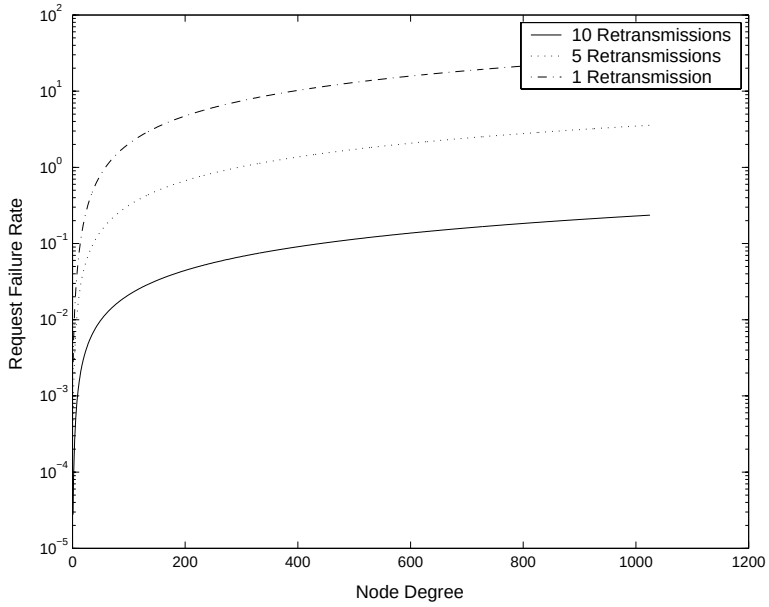

Fig. 13. Analytical Failure Rates for Power-law Networks ($\lambda_0 = 0.95$)

Table I. Analytical Failure Rate Comparison ($\lambda_0 = 0.95$)

Request TTL	Power-law Network (Degree=1)	Fully Connected Network
10	0.0000277	0.000000000672
5	0.000388	0.00000969
1	0.0023	0.0144

Table II. Ratio of Analytical Failure Rates ($\lambda_0 = 0.95$)

Request TTL	Failure Ratio (Power-law/Fully Connected)
10	41220.24
5	40.04
1	0.16

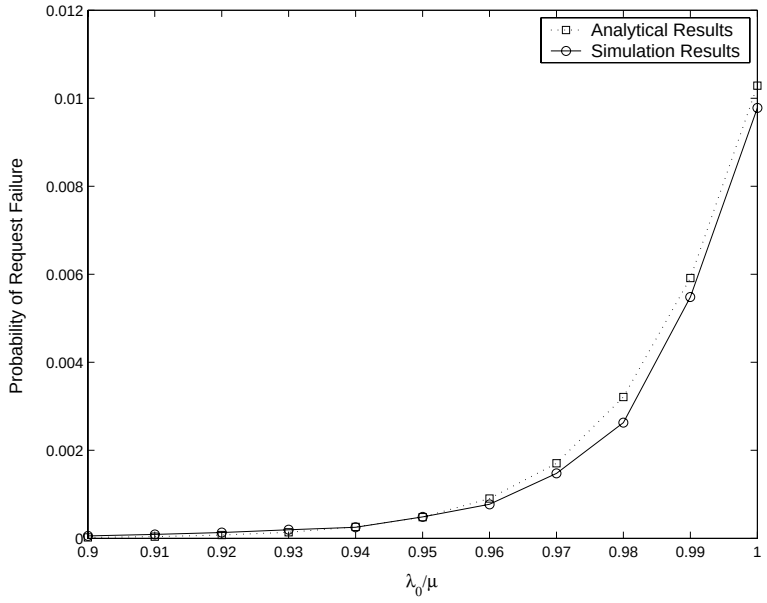

Fig. 14. Power-law Network Failure Probability Comparison (TTL=10)

fully connected network, while the overall failure probability is higher in the power-law network. This shows that highly connected nodes are extremely overwhelmed in this scenario.

2. Analytical Validation

The simulation results which were obtained validate the theoretical model. Figures 14 and 15 compare the theoretical and simulation results for the case of request TTL set to 10. The simulation results match the values obtained by the analytical model.

Figures 16 and 17 show that the simulation results with a TTL of 5 matches the theoretical results as well. The results with a request TTL of 1 is given in Figures 18 and 19. The simulation data for the fully connected network validate the theoretical results. The data for the power-law network depicts a higher failure probability in

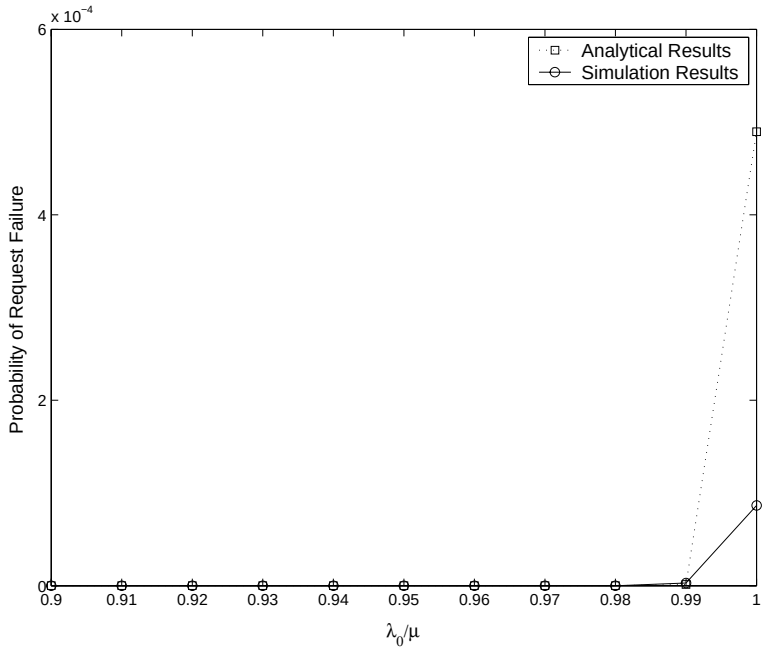

Fig. 15. Fully Connected Network Failure Probability Comparison (TTL=10)

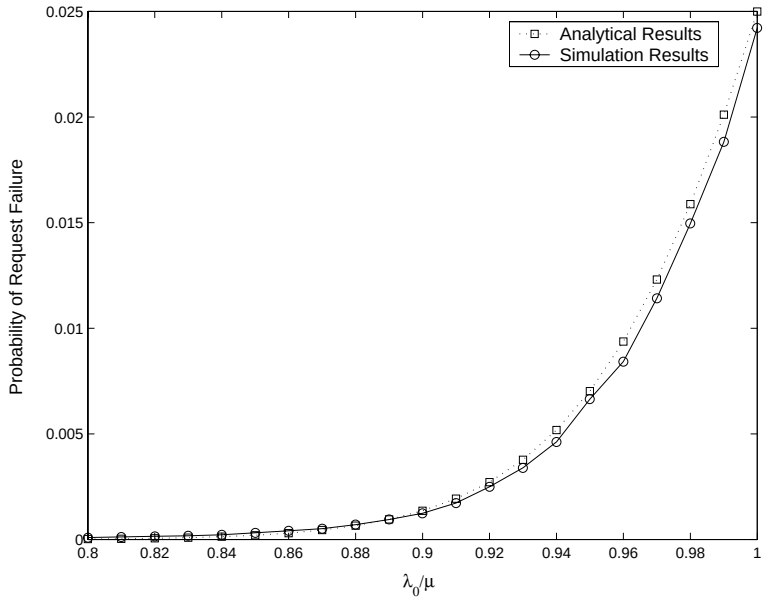

Fig. 16. Power-law Network Failure Probability Comparison (TTL=5)

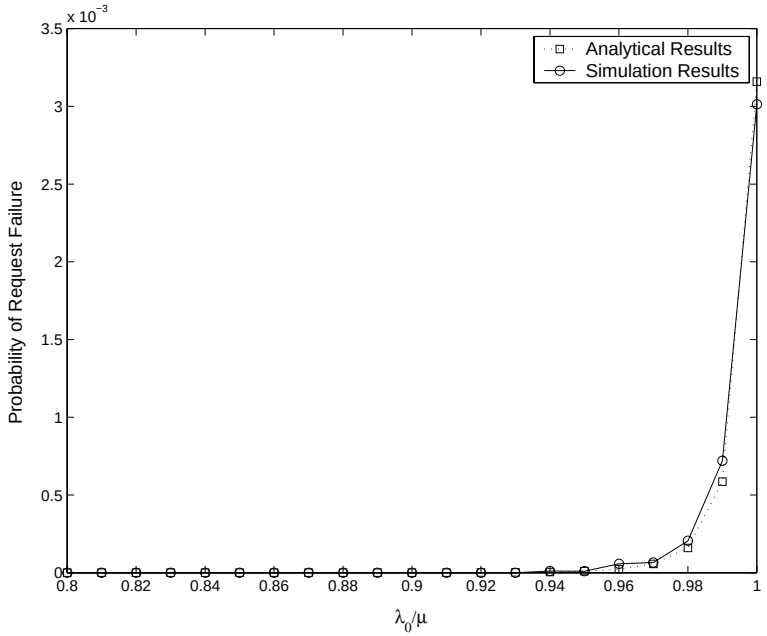

Fig. 17. Fully Connected Network Failure Probability Comparison (TTL=5)

the simulation results than in the analytical results. This result can be explained by recalling the fact that we assumed that the degree of neighboring nodes is independent in equation (3.6).

The assumption was made due to the unavailability of universal correlation statistics for this relationship. But, our simulation topology inherently contains some correlations, including the fact that a node of degree one cannot be the neighbor of another single degree node. When a request can only be forwarded a single time, the majority of forwarded requests will affect highly connected nodes. This effect is magnified when no two single degree nodes can be neighbors. The fact that our theoretical model can allow this situation to occur causes the overall failure probability to decrease since forwarded requests can be spread across more nodes. This accounts for the discrepancy between simulation and analytical results when the request TTL

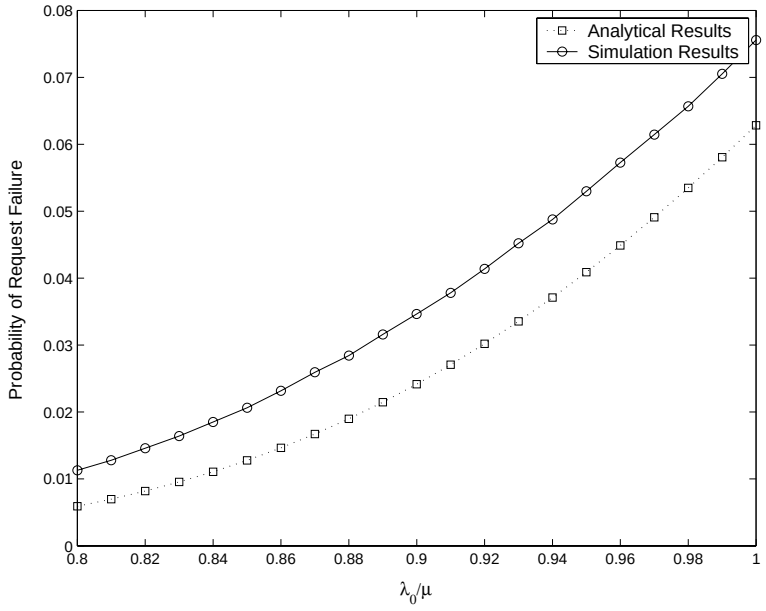

Fig. 18. Power-law Network Failure Probability Comparison (TTL=1)

is one.

Figures 20, 21, and 22 display the failure rate comparisons for the power-law network simulations. Since failure rates increase quickly with decreasing TTL, the fact that the simulation results are on the same order of magnitude as our analytical results validates our failure rate results. It should be noted that in the case of request TTL set to 1, highly connected nodes have a higher failure rate in the simulation results than in the analytical results as expected by our previous arguments.

Table III compares the failure rates observed in the fully connected network scenarios. Once again, the simulation results successfully validate the theoretical results.

Overall, the simulation results that we obtained validate our theoretical model.The relatively small differences that occurred at the boundary case of request TTL set to 1 were identified and explained.

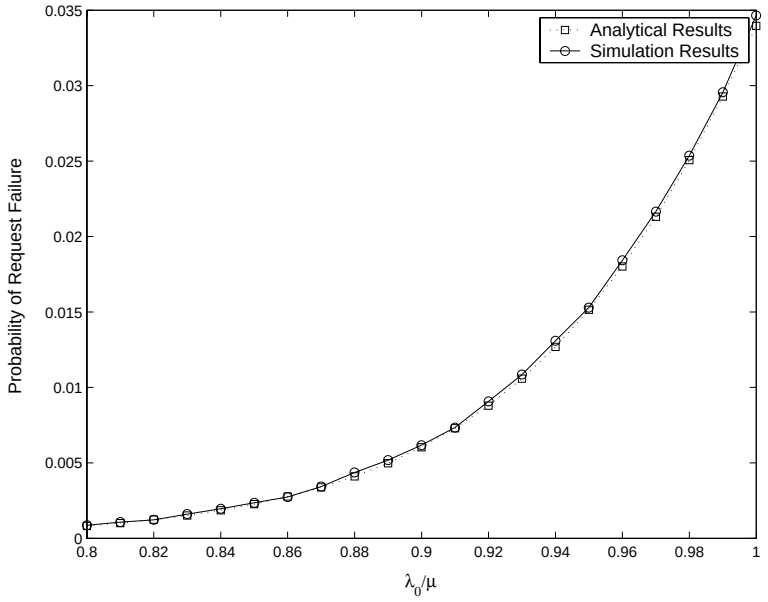

Fig. 19. Fully Connected Network Failure Probability Comparison (TTL=1)

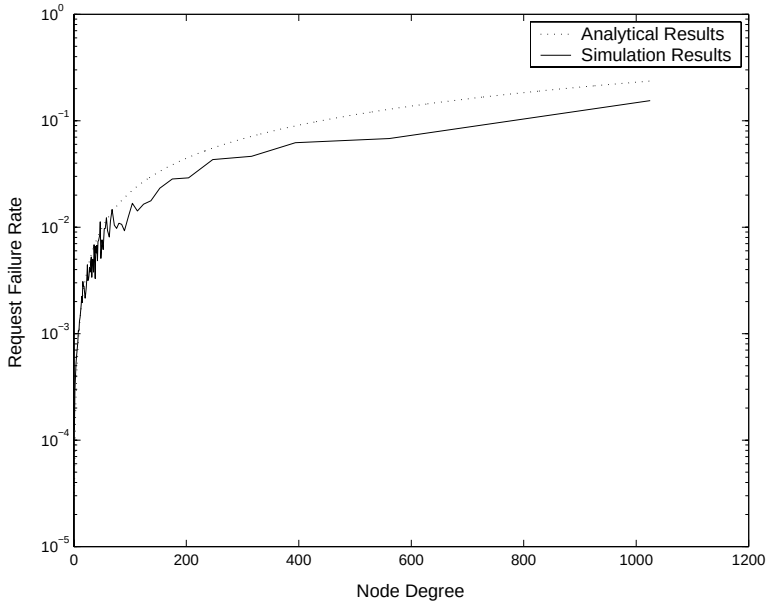

Fig. 20. Power-law Network Failure Rate Comparison(TTL=10)

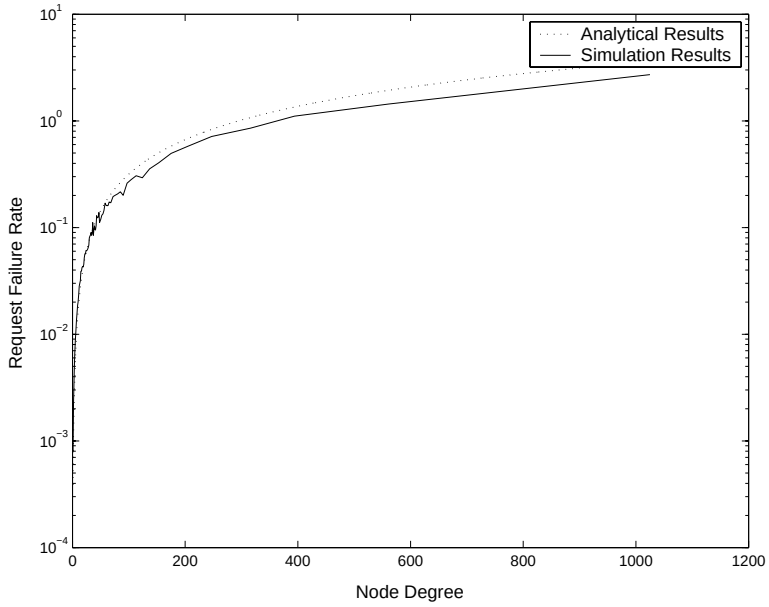

Fig. 21. Power-law Network Failure Rate Comparison(TTL=5)

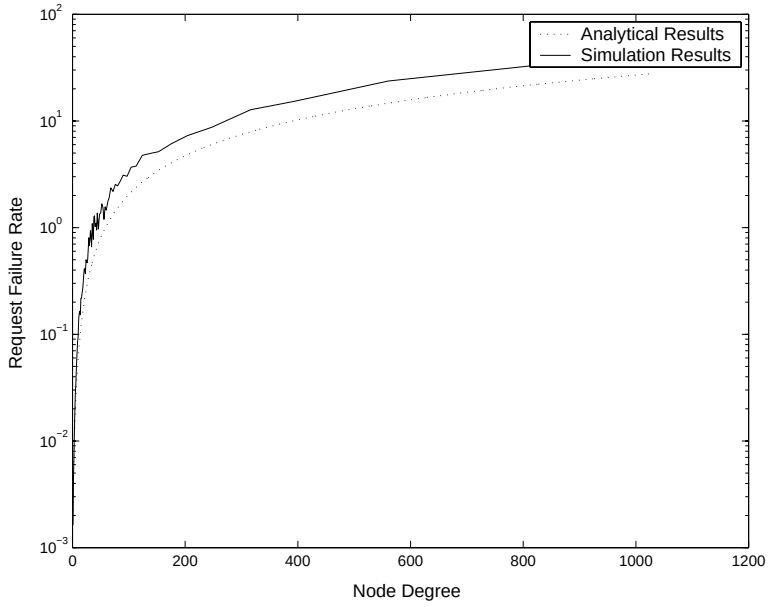

Fig. 22. Power-law Network Failure Rate Comparison(TTL=1)

Table III. Fully Connected Network Failure Rate Comparison ($\lambda_0 = 0.95$)

Request TTL	Analytical Result	Simulation Result
10	0.00000000067	0.0
5	0.00000097	0.000011
1	0.0144	0.0141

3. Effects of Various Power-Laws

For our simulation topologies, the power-law exponent τ was chosen to be 2.1. Though the research that has been cited shows that peer-to-peer systems such as Freenet and Gnutella have power-law distributions in their node connectivity, there is no precise τ that characterizes all of these networks. Figure 23 shows how the failure probability varies with τ when requests can only be forwarded once, and λ_0 is 0.9.

Though the value of τ may not be 2.1 for peer-to-peer systems, it cannot vary much from it. As the figure shows, if τ is slightly smaller, failure probabilities actually increase. If τ is slightly increased, though the failure probability does decrease, it is still much greater than that of the equivalent fully connected network. In fact, τ would have to be very large in order to approach the result for the fully connected network. But if τ is large, there are virtually no highly connected nodes, which measurements have shown is not the case in peer-to-peer networks. Therefore, our conclusions hold for any value of τ which characterizes a particular peer-to-peer network.

C. Lessons Learned

The results that were obtained from our analytical models and simulation data lead us to some important conclusions regarding random algorithms for dynamic resource

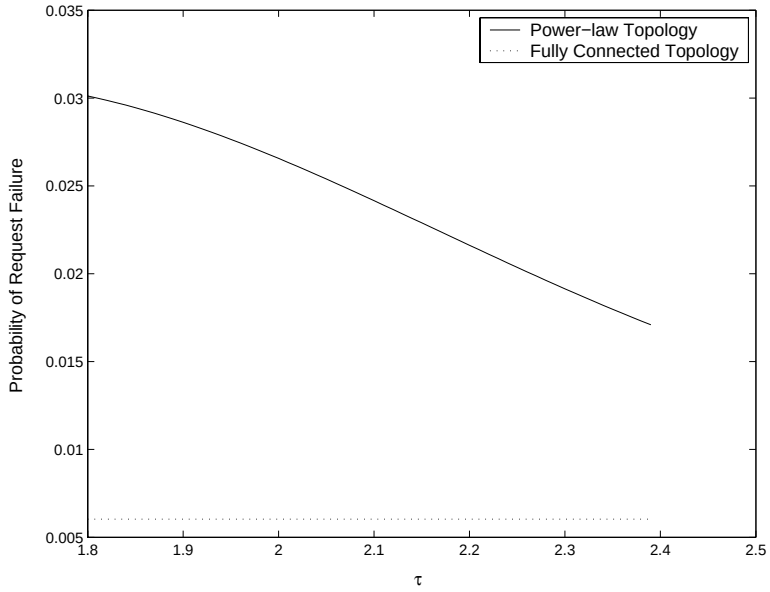

Fig. 23. Power-law Failure Probability with Varying τ (TTL=1, $\lambda_0 = 0.9$))

location in peer-to-peer networks. Our data shows that a fully connected topology provides an improved request failure probability over the power-law distribution which dominates many peer-to-peer systems.

Our results also show that with a fully connected topology, all nodes will experience the same low failure rate. In power-law networks, though, highly connected nodes experience much higher failure rates than poorly connected nodes. Moreover, in most cases all nodes in a power-law distributed network experience a higher failure rate than nodes in a fully connected topology. Thus the use of a hash based overlay network helps provide load balancing for random resource location.

These performance results support the conclusion that the overhead of an overlay network to mimic a fully connected network is worthwhile for random resource location in peer-to-peer networks. Hash based peer-to-peer overlay networks can provide the effect of a fully connected topology with only distributed peer knowledge. There-

fore, these systems should be used as the message routing architecture for random peer-to-peer resource location.

CHAPTER V

A HYBRID APPROACH

In Chapter IV it was concluded that an overlay network is highly beneficial for random resource location in peer-to-peer networks. An overlay network can provide reduced request failure probabilities, as well as improved load balancing when compared to power-law based peer-to-peer systems. The problem with random resource location, though, is that requests may fail even when resources are available in the system. For critical, time sensitive applications, this may not be tolerable.

In this Chapter, we introduce a protocol that can be used along with the peer-to-peer overlay network. It provides some deterministic search capabilities that may be beneficial for sensitive applications. Since our proposed algorithm is designed as a deterministic extension to the random location scheme, our approach is a hybrid system.

A. Resource Location Design Goals for Peer-to-Peer Systems

Grid environments, like peer-to-peer systems, are large scale distributed systems. In [16], the authors present design characteristics for a Grid scale peer-to-peer resource discovery system. One of these is lack of centralized control. The random protocol that we have developed is completely distributed. In order to meet this design constraint, any mechanism introduced to provide determinism must also be distributed.

A second important characteristic for a large scale resource location scheme is to be able to handle resource heterogeneity with respect to requests and to resources offered by nodes. Random resource location does not handle this issue directly. Our proposed extension should therefore address this issue.

A final design consideration is that with the numerous types of resources that

may be available in a large scale system, a global naming scheme may not be viable. Our approach will allow peers which support similar resources to negotiate a naming scheme which will represent the dynamic characteristics of their resources.

B. Deterministic Resource Location

One way to provide deterministic resource location is to have peers aware of the resource availability at other peers. In large systems, this can only be accomplished if subgroups of nodes use this cooperative approach. The benefit of these subgroups is that heterogeneity can be handled easily. Nodes which provide similar resources can join the same cooperative group.

In order to improve the scalability for these subgroups, we remove the restriction that all nodes in a subgroup must be updated when resource availability changes at a single node. This is done by utilizing a peer-to-peer approach inspired by the CAN system. Like CAN, a d-dimensional space is used to organize nodes. In this circumstance, though, the dimensions represent the types of resources that the nodes are cooperating to provide. Instead of being placed randomly in the hyper space, nodes are placed according to the amount of resources available. For example, in the case of two resources, a node is at the point (r_A, r_B), where r_A and r_B are the amount of resource A and B, respectively, that the node has available.

Similar to CAN, this infrastructure requires that nodes keep $2d$ neighbors. These pointers are assigned two per dimension so that nodes are organized with increasing availability in each dimension. Any ties in resource availability amounts are broken by some deterministic tie breaker. Figures 24 and 25 illustrate an example in a two dimensional mesh. This infrastructure can be used to deterministically locate resource tuples in the cooperative group.

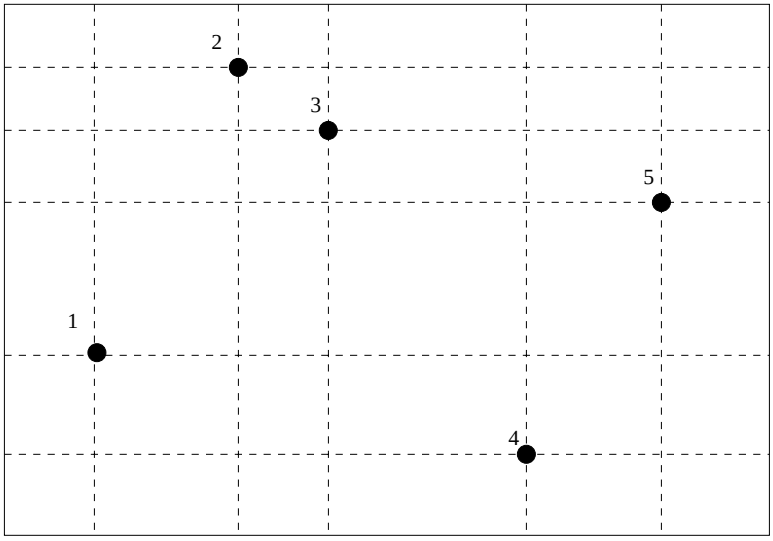

Fig. 24. Peers in Deterministic Overlay Mesh

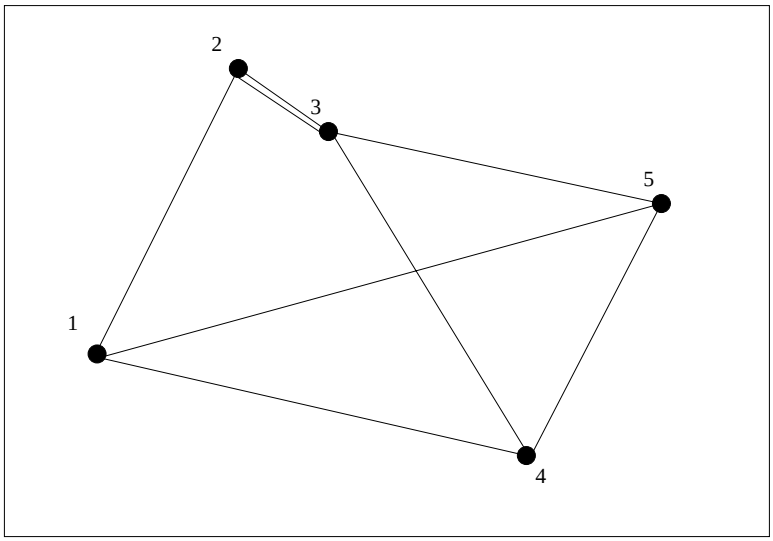

Fig. 25. Peer Connectivity in Deterministic Overlay Mesh

Resource requests can be solved in the d-dimensional space using a simple algorithm. The constraining resource for a given request is the resource which is quantitatively the greatest of the resources that cannot be served by the initial node. Therefore, when the request is being forwarded in the mesh, it should always move towards an increasing value in this dimension. A node can thus forward the request to any neighbor as long as the neighbor has a higher availability of the constraining resource for that request. In this way the request will not encounter a loop, and if the resource tuple is available in the cooperative group, it will be found.

A complication that arises by coordinating nodes according to resource availability is that when availability attributes change at a node, it must migrate in space. This migration can be handled by making a special migration request which locates the new neighbors for the migrating node. A migration request locates new neighbors for a node a dimension at a time. The request is marked for a dimension, and is routed as though that dimension were its constraining resource. The request thereby finds the new neighbors in a dimension. As it is passed along, the request also keeps track of encountered peers which are closest to the migrating node's current resource availability. This information can be used by the migrating node to locate neighbors in other dimensions more quickly.

C. Extending Random Scheme with Deterministic Capabilities

The deterministic approach described in the previous section can be effective when combined with the random resource location scheme. In fact, the random location infrastructure is used by a node to locate nodes for its deterministic subgroup.

Whenever requests are forwarded randomly, a node can send along information regarding its shared resources. Receiving nodes can compare this information to their

own, and make a decision as to whether the candidate node is an appropriate match. Thus subgroup peers can be found in a globally random fashion.

In order to use the two schemes effectively, a simple layered hybrid protocol can be used. As before, requests have some TTL for forwarding. Whenever the request is received by a node which has the correct type of resources, but not enough available, the request can traverse the deterministic layer for that node to attempt to find the required resources. At any point nodes can switch back to the random search mechanism. As before, requests fail when their TTL expires.

The deterministic extension does require added overhead due to the extra peer information and updates that are required. This overhead is not substantial compared to the existing overhead necessary for the overlay network used for random resource location. Also, since peers can choose whether or not to participate in deterministic subgroup(s), they can individually decide whether or not to accept the overhead. The benefit of the extension is that requests can attempt to use the deterministic protocol when system utilization is high. This hybrid approach also increases the ability of nodes to locate rare resource tuples.

CHAPTER VI

SUMMARY AND CONCLUSIONS

Peer-to-peer networks are a plausible infrastructure for large scale systems. Their lack of centralized control and limited state per node make them extremely scalable. Dynamic resource location is an important operation in any computing system. Therefore, it is important to find an effective approach for resource location in large scale peer-to-peer schemes.

Peer-to-peer systems rely on a variety of topologies for message routing. In order to provide an efficient resource location scheme, this characteristic must be taken into account. The research presented in this thesis analyzes the effect of topology on random resource location. In particular, we concentrate on the fully connected topologies obtained by using a hash based overlay network, and power-law distributed topologies which occur in existing peer-to-peer systems. The results of our analysis show that a fully connected topology provides improved request failure probabilities as well as load balancing when compared to power-law distributed systems. The benefits of the overlay network therefore outweigh the overhead of a hash based overlay network.

After reaching and defending our conclusion, we take advantage of the necessary overlay network to provide a deterministic extension to the random resource location infrastructure. This extension uses the fact that random peers will talk to each other during random resource location in order to build cooperative groups which can deterministically locate resource tuples.

REFERENCES

[1] M. Mitzenmacher, "On the analysis of randomized load balancing schemes," in *Proceedings of ACM Symposium on Parallel Algorithms and Architectures*, Newport, RI, June 1997, pp. 292–301.

[2] I. Clarke, O. Sandberg, B. Wiley, and T.W. Hong, "Freenet: A distributed anonymous information storage and retrieval system," in *ICSI Workshop on Design Issues in Anonymity*, Berkeley, CA, July 2001, pp. 46–66.

[3] Clip2.com, "The Gnutella protocol specification v0.4," 2000, http://www9.limewire.com/developer/gnutella_protocol_0.4.pdf (accessed February 13, 2003).

[4] M. Ripeanu, "Peer-to-peer architecture case study: Gnutella network," in *Proceedings of International Conference on Peer-to-Peer Computing*, Linkoping, Sweden, August 2001, pp. 99–100.

[5] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," in *Proceedings of ACM SIGCOMM*, San Diego, CA, August 2001, pp. 149–160.

[6] A. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems," in *Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms*, Heidelberg, Germany, November 2001, pp. 329–350.

[7] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content-addressable network," in *Proceedings of ACM SIGCOMM*, San Diego, CA, August 2001, pp. 161–172.

[8] M. Balazinska, H. Balakrishnan, and D. Karger, "Ins/Twine: A scalable peer-to-peer architecture for intentional resource discovery," in *International Conference on Pervasive Computing*, Zurich, Switzerland, August 2002, pp. 195–210.

[9] W. Adjie-Winoto, E. Schwartz, H. Balakrishnan, and J. Lilley, "The design and implementation of an intentional naming system," in *Symposium on Operating Systems Principles*, Charleston, SC, December 1999, pp. 186–201.

[10] M. Faloutsos, P. Faloutsos, and C. Faloutsos, "On power-law relationships of the internet topology," in *Proceedings of ACM SIGCOMM*, Cambridge, MA, August 1999, pp. 251–262.

[11] L. Adamic, R. Lukose, A. Puniyani, and B. Huberman, "Search in power-law networks," *Physical Review E*, vol. 64, pp. 46135–46143, October 2001.

[12] S. Sen and J. Wang, "Analyzing peer-to-peer traffic across large networks," in *Internet Measurement Workshop*, Marseille, France, November 2002, pp. 137–150.

[13] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker, "Search and replication in unstructured peer-to-peer networks," 2001, http://citeseer.nj.nec.com/lv02search.html (accessed February 11, 2003).

[14] University of Newcastle Upon Tyne, "Javasim: User's guide," http://javasim.ncl.ac.uk/manual/javasim.pdf (accessed February 17, 2003).

[15] J. Winick and S. Jamin, "Inet-3.0: Internet topology generator," http://topology.eecs.umich.edu/inet/inet-3.0.pdf (accessed February 17, 2003).

[16] A. Iamnitchi, I. Foster, and D. Nurmi, "A peer-to-peer approach to resource discovery in grid environments," in *IEEE International Symposium on High*

Performance Distributed Computing, Edinburgh, Scotland, July 2002, p. 419.

Supplemental Sources Consulted

D. Bertsekas and R. Gallager, *Data Networks*, 2nd edition, Upper Saddle River, NJ: Prentice Hall, 1992.

D. Tang, K. Chang, C.and Tanaka, and M. Baker, "Resource discovery in ad hoc networks," Tech. Rep. CSL-TR-98-769, Stanford University, Stanford, CA, August 1998.

K. Trivedi, *Probability and Statistics with Reliability , Queuing and Computer Science Applications*, 2nd edition, New York, NY: Wiley-Interscience, 2002.

VITA

Ripal Babubhai Nathuji currently resides at 6503 Pecan Acres Drive in Leon Valley, TX 78240. He received his B.S. in electrical engineering and computer science from the Massachusetts Insitute of Technology in May, 2001.

The typist for this thesis was Ripal Babubhai Nathuji.